Short course summary

The course can more or less be divided up into 3 parts :

· Enumerative combinatorics
· Arithmetic
· Graph theory

Enumerative Combinatorics

This involved studying more or less sophisticated techniques for 'counting', that is, for computing integer-valued functions $f(n)$ of an integer variable n. We introduced three broad techniques

(a) Multiplication principle (MP). This simple principle has applications to counting, for example,

· ordered selections with repitition allowed

· ordered selections with repitition not allowed, of which an important special case is *permutations*

· unordered selections with repitition allowed (godisar till barn, or, more formally, placing indistinguishable objects in distinguishable cells)

· unordered selections with repitition not allowed, so-called *combinations*. The binomial coefficient $C(n, k)$ is the number of ways to choose k different objects from n. We learned some identities involving binomial coefficients, in particular the *binomial theorem*.

(b) Recurrence relations.

If you can't directly (i.e.: using something as simple as, say, MP) find an explicit formula for a function $f(n)$, the next-best thing is often to look for a recurrence relation. To find such a relation involves counting your $f(n)$ objects in a smart way, usually by dividing them up into a small number of 'types'. Sometimes the recurrence relation can be 'solved' to produce an explicit formula for $f(n)$.

(i) We illustrated how one finds recurrence relations by giving several of the more famous examples (as well as some not so famous ones) : Fibonacci numbers, Stirling numbers of the second kind $S(n, k)$, integer partitions $p(n, k)$ and Catalan numbers.

(ii) We developed general techniques for solving *linear recurrence relations with constant coefficients.* In the *homogeneous* case, one just needs to solve an auxiliary polynomial equation. In the non-homogeneous case, one introduces a so-called *generating function.* The method of generating functions can, in theory, be applied to any recurrence relation whatsoever. It rarely leads anywhere,[1], though the cases in which it does so are important. As well as the non-homogeneous linear recurrences, we illustrated how the generating function method yields an explicit formula for the Catalan numbers.

One important technical tool in applying the GF method was a *generlised binomial theorem.*

(c) Inclusion-Exclusion (Sieve) Principle.

This is a very primitive counting technique of limited application. A classic example of its' applicability is to counting *derangements.* Other applications appear in the exercises.

ARITHMETIC

There were two main (and sometimes interlinked, for example in the discussion of *RSA cryptography*) themes

(a) Integer factorisation.

The *Fundamental Theorem of Arithmetic*, proven by Euclid in around 300BC, is the central result here. In Euclid's treatment, the basic concept is that of the *greatest common divisor* of two integers. *Euclid's algorithm* gives a very fast method for computing this.

(b) Modular arithmetic.

We defined the notion of *congruence* modulo an integer n. The central result is the following : for a given n, the relation of congruence modulo n is an *equivalence relation* on the integers $\mathbf{Z}$. There are n equivalence classes,

[1] There is the analogous *power series method* for tackling ordinary differential equations. Once again it is, in theory, always applicable, but only in rare, though important cases, yields results.

and they form a *ring* (notation : $\mathbf{Z}/n\mathbf{Z}$) under addition and multiplication modulo n.

Among the more advanced results, the most famous is probably *Fermat's (little) theorem* and its' generalisation by Euler. The simplest application of these ideas is to fast computation of $a^b \pmod{c}$.

Remark Much of the theory described above was and is motivated by the desire to develop techniques for studying *Diophantine equations.* These are ordinary polynomial equations, but where all variables are considered as integer-valued. We mostly studied *linear* Diophantine equations

$$ax + by = c$$

where Euclid'a algorithm yields a general method of solution. There exists a general theory for *binary (i.e.: 2-variable) quadratic* equations, developed by Gauss, which we only hinted at in one homework exercise. Other than that, we only had scattered examples.

One common way of proving that a Diophantine equation has no solution is to show that it has no solution modulo a certain integer n. This leads us to also study *Diophantine congruences.* In the absence of anything better, one can always try to solve a Diophantine congruence by *exhaustive search*, since there are only finitely many possibilities for each variable. In some cases, we saw how to do better :

· Euclid's algorithm gives a method to solve a 1-variable linear congruence

$$ax \equiv b \pmod{n}.$$

· The *Chinese Remainder Theorem* gives an algorithm to solve a system of 1-variable congruences, provided the bases are relatively prime.

· The formula for the roots of a quadratic equation (or the method of completing squares, which is the same thing) reduces the amount of searching needed to solve quadratic 1-variable congruences (see ex. 1 for Thursday, week 4).

· Sometimes Fermat's or Euler's theorem can be applied to reduce the amount of searching required (see ex. 4 for Friday, Week 3 and Q.6 on 3rd practice exam)

· In general, when solving $p(x) \equiv 0 \pmod{n}$, it reduces the workload if you can factorise $p(x)$ and/or n (this follows from FTA). See, for example, Q.4 on 3rd practice exam).

Graph Theory

We discussed the following topics :

(a) Euler paths and cycles. *Euler's theorem* gives a necessary and sufficient condition for a graph to have an Euler path/cycle. If the conditions are satisfied, a suitable path/cycle can be found by DFS.

(b) Ramsey numbers. $R(p, q)$ is the least integer n such that every simple graph on n vertices contains either a *clique* of size p or an *independent set* of size q. Not much is known about Ramsey numbers, but in a homework exercise we proved that $R(4, 3) = 10$.

(c) Hamilton paths/cycles. The problem of determining whether a graph has a Hamilton path/cycle (so-called *travelling salesman problem*) is NP-complete. *Dirac's theorem* gives a (very restrictive) sufficient condition. *Petersen's graph* is a famous example of a graph without a Hamilton cycle.

(d) Graph coloring. The problem of finding the *chromatic number* of a graph is NP-complete. The *greedy algorithm* is a general graph-coloring algorithm. A graph is 2-colorable (so-called *bipartite*) if and only if it has no odd cycles. The famous *four color theorem* states that every planar graph is 4-colorable. As an aside, in a homework exercise you proved *Euler's formula* for planar graphs : $V - E + R = 1$.

(e) Trees. Spanning trees, minimal spanning trees (MST) in weighted graphs, shortest paths in weighted (di)graphs. A spanning tree can be found by DFS, but finding a minimal spanning teee requires BFS. *Dijkstra's algorithm* for finding a shortest path is also of BFS type.